Lógica

Resolución con *UMG*

Damiano Zanardini

Graduado/a en Ingeniería Informática
Facultad de Informática
Universidad Politécnica de Madrid
damiano@fi.upm.es

Curso Académico 2010/2011

Resolución con UMG

Regla de resolución con *UMG*

Sean

$$C_1 = L_1 \vee .. \vee L_n \vee D_1 \qquad\qquad C_2 = \neg L_{n+1} \vee .. \vee \neg L_{n+m} \vee D_2$$

dos cláusulas, donde todos los literales L tienen el mismo símbolo de predicado. Se puede deducir una nueva cláusula

$$(D_1\rho \vee D_2)\theta$$

donde

- ρ es un renombrado tal que $C_1\rho$ y C_2 no tienen ningún nombre de variable en común
- θ es el *UMG* de $L_1\rho, \ldots, L_n\rho, L_{n+1}, \ldots, L_{n+m}$

La nueva cláusula se llama el *resolvente* de C_1 y C_2

Resolución con UMG

Regla de factorización

- dada una cláusula $C = L_1 \vee .. \vee L_n \vee D$, donde L_i tienen el mismo símbolo de predicado, se puede deducir una nueva cláusula $C' = L \vee D\alpha$ donde
  - α es un unificador (no necesariamente el *UMG*) de $L_1, .., L_n$
  - $L = L_1\alpha = .. = L_n\alpha$
- L se llama *factor* de $L_1 \vee .. \vee L_n \vee D$

☞ se note que la nueva cláusula no es nada más que una *instancia* de la vieja, obtenida aplicando la sustitución α

☞ entonces la nueva cláusula representa un hecho lógico *menos general* que la vieja

☞ en otra palabras, factorizando nos olvidamos de una parte de la información

Resolución con UMG

Paso de *Resolución con UMG*

Aplicación optativa de la regla de factorización, seguida por la resolución con UMG

- in the system described in this paper, one such inference principle is used. It is called the *resolution principle*, and it is machine-oriented, rather than human-oriented [R65]

☞ se note que la factorización es optativa: los resolventes se pueden generar con o sin ella

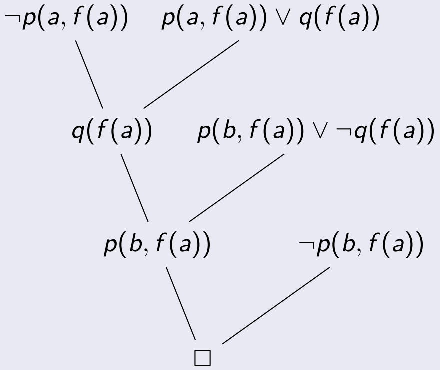
El método

Se pueden construir árboles de resolución en los que los resolventes de cada dos cláusulas se obtienen en un paso de resolución

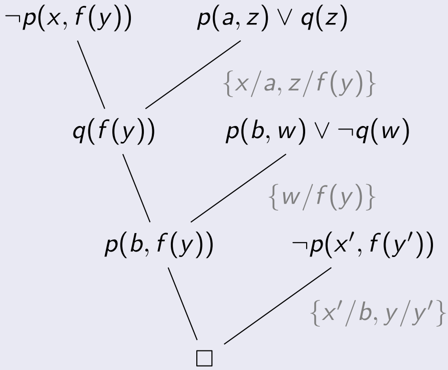
- por cada paso de *resolución en instancias básicas* hay un paso de *resolución con UMG*

Resolución con UMG

$C_1 = \neg p(x, f(y)),\ C_2 = p(a, z) \vee q(z),\ C_3 = p(b, u) \vee \neg q(u)$



resolución con instancias básicas



resolución con UMG

Resolución con UMG

Lema (Regla de resolución con *UMG*, corrección)

$[\ \forall x_1..x_p C_1, \ \ \forall y_1..y_q C_2\ ] \quad \vdash \quad \forall z_1..z_r((D_1\rho \vee D_2)\theta)$ *es correcta, donde*

- $\{x_1,..,x_p\} = Vars(C_1)$, $\{y_1,..,y_q\} = Vars(C_2)$, $\{z_1,..,z_r\} = Vars((D_1\rho \vee D_2)\theta)$
- *ρ es un renombrado de $x_1..x_p$ definido como arriba*
- $\theta = UMG(L_1\rho, \ldots, L_n\rho, L_{n+1}, \ldots, L_{n+m})$

Demostración.

❶	$\forall x_1..x_p(L_1 \vee .. \vee L_n \vee D_1)$	hipótesis ($C_1 = \overline{L} \vee D_1$)
❷	$\forall z_1..z_r(\neg L_{n+1} \vee .. \vee \neg L_{n+m} \vee D_2)$	hipótesis ($C_2 = \overline{\neg L} \vee D_2$)
❸	$F \vee E_1$	aplicación de ρ y θ a C_1, idempotencia $F \vee .. \vee F = F$
❹	$\neg F \vee E_2$	aplicación de θ a C_2, idempotencia $\neg F \vee .. \vee \neg F = \neg F$
❺	$E_1 \vee E_2$	corte sobre ❸ y ❹
❻	$\forall z_1..z_r((D_1\rho \vee D_2)\theta)$	generalización de ❺

Resolución con UMG

Una observación importante

☞ la regla de resolución con *UMG no implica que hayan tenido lugar todos los posibles pasos de factorización*

- *de hecho la factorización puede ayudar en algunos casos, pero también puede hacer la derivación imposible*

$$
\begin{array}{ll}
C_1 & \neg p(x,y,u) \vee \neg p(y,z,v) \vee \neg p(u,z,w) \vee p(x,v,w) \\
C_2 & \neg p(x,y,u) \vee \neg p(v,z,y) \vee \neg p(x,v,w) \vee p(u,z,w) \\
C_3 & p(x,e,x) \\
C_4 & p(x,i(x),e) \\
C_5 & p(i(x),x,e) \\
C_6 & \neg p(c,c,e)
\end{array}
$$

Resolución con UMG

Una observación importante

☞ la regla de resolución con *UMG no implica que hayan tenido lugar todos los posibles pasos de factorización*

- *de hecho la factorización puede ayudar en algunos casos, pero también puede hacer la derivación imposible*

$$\begin{array}{ll}
C_1 & \neg p(x,y) \vee \neg q(f(y),x) \\
C_2 & p(x,g(x)) \\
C_3 & r(x,a) \vee \neg p(b,g(x)) \vee r(z,z) \\
C_4 & q(f(g(x)),a) \vee \neg r(x,a) \vee \neg r(a,y) \\
C_5 & p(x,g(y))
\end{array}$$

Resolución con UMG

Teorema (resolución con *UMG*)

Un conjunto $\mathcal{C}$ de clásusulas es insatisfacible sii se puede deducir $\Box$ por resolución con UMG ($\mathcal{C} \vdash_{UMG} \Box$)

Método de Saturación

Sea $\mathcal{C}$ un conjunto de cláusulas

$S_0 = \mathcal{C}$
$n = 0$
repite
  si $(\square \in S_n)$ **entonces STOP**: $INSAT(\mathcal{C})$
  en otro caso
    $S_{n+1} = \{\text{resolventes de } C_1 \text{ y } C_2 \mid C_1 \in S_1 \cup .. \cup S_n,\ C_2 \in S_n\}$
    si $(S_{n+1} = \emptyset)$ o $(S_{n+1} \subseteq S_1 \cup .. \cup S_n)$ **entonces STOP**: $SAT(\mathcal{C})$
    $n = n + 1$

Completitud: $INSAT(\mathcal{C})$ sii se deduce $\square$

- la construcción de S_{n+1} necesita que consideremos todos los posibles factores de C_1 y C_2
- este método genera *todos* y *sólo* los resolventes de las cláusulas de $\mathcal{C}$
- se generan muchas cláusulas redundantes

Método de Saturación

Ejemplo: $\mathcal{C} = \{p \vee q,\ \neg p \vee q,\ p \vee \neg q,\ \neg p \vee \neg q\}$

$$S_0 = \begin{array}{rl} (1) & p \vee q \\ (2) & \neg p \vee q \\ (3) & p \vee \neg q \\ (4) & \neg p \vee \neg q \end{array} \qquad S_1 = \begin{array}{rll} (5) & q & (1,2) \\ (6) & p & (1,3) \\ (7) & q \vee \neg q & (1,4) \\ (8) & p \vee \neg p & (1,4) \\ (9) & q \vee \neg q & (2,3) \\ (10) & p \vee \neg p & (2,3) \\ (11) & \neg p & (2,4) \\ (12) & \neg q & (3,4) \end{array}$$

ya después de un paso hay cláusulas redundantes y tautologías

Método de Saturación

Conclusión

- la resolución *UMG* permite decidir la satisfacibilidad sin tener que considerar las instancias básicas
- pero la saturación no es optimal porque genera muchas cláusulas inútiles
  - the raw implementation of the Resolution Principle would produce a very inefficient refutation procedure [R65]
  - by Church's Theorem we know that for some inputs S this procedure, and in general all correct refutation procedures, will not terminate [R65]

Ejemplo [R65]

$$C_1 = q(a) \qquad\qquad C_2 = \neg q(x) \vee q(f(x))$$

en cada paso se genera $q(f^n(a))$ con n que aumenta de 1 cada vez